Search

How to access serial and parallel ports by using Visual Basic .NET

For a Microsoft Visual Studio 2005 version of this article, see 904795 (http://support.microsoft.com/kb/904795/).

Article ID : 823179
Last Review : November 26, 2007
Revision : 3.4

On This Page

SUMMARY

This step-by-step article describes how to access serial ports and how to access parallel ports by using Microsoft Visual Basic .NET. This article also contains sample code that illustrates the concepts that are discussed in this article.

Requirements

The following list outlines the recommended hardware, software, network infrastructure, and service packs that you need:

- Microsoft Windows Server 2003, Microsoft Windows XP, or Microsoft Windows 2000
- Visual Basic .NET

This article assumes that you are familiar with the following topics:

- Programming in Visual Basic .NET
- Platform invoke services in Visual Basic .NET

Use the MSComm control in Visual Basic .NET to access serial ports

Because no Microsoft .NET Framework classes exist to access the communications resources that connected to your computer, you can use the **MSComm** control in Microsoft Visual Basic 6.0. The **MSComm** control provides serial communications for your application by enabling the transmission and reception of data through a serial port. To implement basic serial communications by using a modem, follow these steps:

1. Start Microsoft Visual Studio .NET.
2. On the **File** menu, point to **New**, and then click **Project**.
3. Under **Project Types**, click **Visual Basic Projects**.
4. Under **Templates**, click **Console Application**.
5. In the **Name** box, type **MyConsoleApplication**, and then click **OK**.

   By default, Module1.vb is created.
6. Right-click the **MyConsoleApplication** project, and then click **Add Reference**.
7. Click the **COM** tab, click **Microsoft Comm Control 6.0** under **Component Name**, click **Select**, and then click **OK**.

   Note To use the **MSComm** control, you must install the related COM components of Microsoft Visual Basic 6.0 on the same computer that has Microsoft Visual Studio .NET installed.

For more information about license issues when you use Visual Basic 6.0 controls in Visual Studio .NET, click the following article number to view the article in the Microsoft Knowledge Base:

318597 (http://support.microsoft.com/kb/318597/) Errors when you use Visual Basic 6.0 controls in Visual Studio .NET

8. Replace the code in Module1.vb with the following code example.

```
Imports MSCommLib

Module Module1

    Sub Main()
        'New a MSComm control
        Dim MSComm1 As MSComm
        MSComm1 = New MSComm
        ' Buffer to hold input string.
        Dim Buffer As String
        ' Use the COM1 serial port.
        MSComm1.CommPort = 1
        ' 9600 baud, no parity, 8 data, and 1 stop bit.
        MSComm1.Settings = "9600,N,8,1"
        ' Tell the control to read the whole buffer when Input is used.
        MSComm1.InputLen = 0
        ' Open the serial port.
        MSComm1.PortOpen = True
        Console.WriteLine("Open the serial port.")
        ' Tell the control to make the Input property return text data.
        MSComm1.InputMode() = InputModeConstants.comInputModeText
        'Clear the receive buffer.
        MSComm1.InBufferCount() = 0
        ' Send the attention command to the modem.
        MSComm1.Output = "ATV1Q0" & Chr(13)
        Console.WriteLine("Send the attention command to the modem.")
        Console.WriteLine("Wait for the data to come back to the serial port...")
        ' Make sure that the modem responds with "OK".
        ' Wait for the data to come back to the serial port.
        Do
            Buffer = Buffer & MSComm1.Input
        Loop Until InStr(Buffer, "OK" & vbCrLf)
        ' Read the "OK" response data in the serial port.
        ' Close the serial port.
        Console.WriteLine("Read the OK response data in the serial port.")
        MSComm1.PortOpen = False
        Console.WriteLine("Close the serial port.")
    End Sub

End Module
```

9. Press CRTL+F5 to build and run this project. You will receive the following output messages:

```
Open the serial port.
Send the attention command to the modem.
Wait for data to come back to the serial port...
Read the OK response data in the serial port.
Close the serial port.
```

Use platform invoke services to call Win32 API functions in Visual Basic .NET to access serial and parallel ports

To do this, follow these steps:

1. Start Microsoft Visual Studio .NET.
2. On the **File** menu, point to **New**, and then click **Project**.
3. Under **Project Types**, click **Visual Basic Projects**.
4. Under **Templates**, click **Console Application**.

5. In the **Name** text box, type **MyConsoleApplication**, and then click **OK**.

   By default, Module1.vb is created.
6. Add the following code to Module1.vb before the **Module Module1** statement:

```
Option Strict On

' Define a CommException class that inherits from the ApplicationException
class,
' and then throw an object of type CommException when you receive an error
message.
Class CommException
   Inherits ApplicationException
   Sub New(ByVal Reason As String)
      MyBase.New(Reason)
   End Sub
End Class
```

7. Declare structures, constants, and references to external functions that are in Kernel32.dll

   To call unmanaged functions from your managed Visual Basic .NET application, you must declare references to the structures that you pass as parameters to the unmanaged functions, and you must declare the constants that you pass as parameters to the unmanaged functions. To do this, add the following code to Module1.vb after the **Module Module1** statement:

```
'Declare structures.
   Public Structure DCB
   Public DCBlength As Int32
   Public BaudRate As Int32
   Public fBitFields As Int32 'See Comments in Win32API.Txt
   Public wReserved As Int16
   Public XonLim As Int16
   Public XoffLim As Int16
   Public ByteSize As Byte
   Public Parity As Byte
   Public StopBits As Byte
   Public XonChar As Byte
   Public XoffChar As Byte
   Public ErrorChar As Byte
   Public EofChar As Byte
   Public EvtChar As Byte
   Public wReserved1 As Int16 'Reserved; Do Not Use
End Structure

Public Structure COMMTIMEOUTS
   Public ReadIntervalTimeout As Int32
   Public ReadTotalTimeoutMultiplier As Int32
   Public ReadTotalTimeoutConstant As Int32
   Public WriteTotalTimeoutMultiplier As Int32
   Public WriteTotalTimeoutConstant As Int32
End Structure

'Declare constants.
Public Const GENERIC_READ As Int32 = &H80000000
Public Const GENERIC_WRITE As Int32 = &H40000000
Public Const OPEN_EXISTING As Int32 = 3
Public Const FILE_ATTRIBUTE_NORMAL As Int32 = &H80
Public Const NOPARITY As Int32 = 0
Public Const ONESTOPBIT As Int32 = 0

'Declare references to external functions.
Public Declare Auto Function CreateFile Lib "kernel32.dll" _
   (ByVal lpFileName As String, ByVal dwDesiredAccess As Int32, _
      ByVal dwShareMode As Int32, ByVal lpSecurityAttributes As IntPtr, _
         ByVal dwCreationDisposition As Int32, ByVal dwFlagsAndAttributes As
Int32, _
            ByVal hTemplateFile As IntPtr) As IntPtr

Public Declare Auto Function GetCommState Lib "kernel32.dll" (ByVal nCid As
IntPtr, _
```

```
   ByRef lpDCB As DCB) As Boolean

Public Declare Auto Function SetCommState Lib "kernel32.dll" (ByVal nCid As
IntPtr, _
   ByRef lpDCB As DCB) As Boolean

Public Declare Auto Function GetCommTimeouts Lib "kernel32.dll" (ByVal hFile As
IntPtr, _
   ByRef lpCommTimeouts As COMMTIMEOUTS) As Boolean

Public Declare Auto Function SetCommTimeouts Lib "kernel32.dll" (ByVal hFile As
IntPtr, _
   ByRef lpCommTimeouts As COMMTIMEOUTS) As Boolean

Public Declare Auto Function WriteFile Lib "kernel32.dll" (ByVal hFile As
IntPtr, _
   ByVal lpBuffer As Byte(), ByVal nNumberOfBytesToWrite As Int32, _
      ByRef lpNumberOfBytesWritten As Int32, ByVal lpOverlapped As IntPtr) As
Boolean

Public Declare Auto Function ReadFile Lib "kernel32.dll" (ByVal hFile As
IntPtr, _
   ByVal lpBuffer As Byte(), ByVal nNumberOfBytesToRead As Int32, _
      ByRef lpNumberOfBytesRead As Int32, ByVal lpOverlapped As IntPtr) As
Boolean

Public Declare Auto Function CloseHandle Lib "kernel32.dll" (ByVal hObject As
IntPtr) As Boolean
```

8. Before you can access a serial port or a parallel port, you must obtain a handle to the appropriate port and then configure the port communications. To do this, add the following initialization code to Module1.vb after the **Sub Main** statement.

   Note To establish communication with the LPT*x* port, you must stop the Print Spooler service. To do this, use the Services tool in Administrative Tools.

```
' Declare the local variables that you will use in the code.
Dim hSerialPort, hParallelPort As IntPtr
Dim Success As Boolean
Dim MyDCB As DCB
Dim MyCommTimeouts As COMMTIMEOUTS
Dim BytesWritten, BytesRead As Int32
Dim Buffer() As Byte

' Declare the variables to use for encoding.
Dim oEncoder As New System.Text.ASCIIEncoding
Dim oEnc As System.Text.Encoding = oEncoder.GetEncoding(1252)

' Convert String to Byte().
Buffer = oEnc.GetBytes("Test")
     Try
         ' Access the serial port.
         Console.WriteLine("Accessing the COM1 serial port")
         ' Obtain a handle to the COM1 serial port.
         hSerialPort = CreateFile("COM1", GENERIC_READ Or GENERIC_WRITE, 0,
IntPtr.Zero, _
            OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, IntPtr.Zero)
         ' Verify that the obtained handle is valid.
         If hSerialPort.ToInt32 = -1 Then
            Throw New CommException("Unable to obtain a handle to the COM1
port")
         End If
         ' Retrieve the current control settings.
         Success = GetCommState(hSerialPort, MyDCB)
         If Success = False Then
            Throw New CommException("Unable to retrieve the current control
settings")
         End If
         ' Modify the properties of the retrieved DCB structure as appropriate.
```

```
            ' WARNING: Make sure to modify the properties according to their
supported values.
            MyDCB.BaudRate = 9600
            MyDCB.ByteSize = 8
            MyDCB.Parity = NOPARITY
            MyDCB.StopBits = ONESTOPBIT
            ' Reconfigure COM1 based on the properties of the modified DCB
structure.
            Success = SetCommState(hSerialPort, MyDCB)
            If Success = False Then
               Throw New CommException("Unable to reconfigure COM1")
            End If
            ' Retrieve the current time-out settings.
            Success = GetCommTimeouts(hSerialPort, MyCommTimeouts)
            If Success = False Then
               Throw New CommException("Unable to retrieve current time-out
settings")
            End If
            ' Modify the properties of the retrieved COMMTIMEOUTS structure as
appropriate.
            ' WARNING: Make sure to modify the properties according to their
supported values.
            MyCommTimeouts.ReadIntervalTimeout = 0
            MyCommTimeouts.ReadTotalTimeoutConstant = 0
            MyCommTimeouts.ReadTotalTimeoutMultiplier = 0
            MyCommTimeouts.WriteTotalTimeoutConstant = 0
            MyCommTimeouts.WriteTotalTimeoutMultiplier = 0
            ' Reconfigure the time-out settings, based on the properties of the
modified COMMTIMEOUTS structure.
            Success = SetCommTimeouts(hSerialPort, MyCommTimeouts)
            If Success = False Then
               Throw New CommException("Unable to reconfigure the time-out
settings")
            End If
            ' Write data to COM1.
            Console.WriteLine("Writing the following data to COM1: Test")
            Success = WriteFile(hSerialPort, Buffer, Buffer.Length, BytesWritten,
IntPtr.Zero)
            If Success = False Then
               Throw New CommException("Unable to write to COM1")
            End If
            ' Read data from COM1.
            Success = ReadFile(hSerialPort, Buffer, BytesWritten, BytesRead,
IntPtr.Zero)
            If Success = False Then
               Throw New CommException("Unable to read from COM1")
            End If
         Catch ex As Exception
            Console.WriteLine(Ex.Message)
         Finally
            ' Release the handle to COM1.
            Success = CloseHandle(hSerialPort)
            If Success = False Then
               Console.WriteLine("Unable to release handle to COM1")
            End If
         End Try

         Try
            ' Parallel port.
            Console.WriteLine("Accessing the LPT1 parallel port")
            ' Obtain a handle to the LPT1 parallel port.
            hParallelPort = CreateFile("LPT1", GENERIC_READ Or GENERIC_WRITE, 0,
IntPtr.Zero, _
               OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, IntPtr.Zero)
            ' Verify that the obtained handle is valid.
            If hParallelPort.ToInt32 = -1 Then
               Throw New CommException("Unable to obtain a handle to the LPT1
port")
            End If
         ' Retrieve the current control settings.
         Success = GetCommState(hParallelPort, MyDCB)
         If Success = False Then
            Throw New CommException("Unable to retrieve the current control
settings")
         End If
         ' Modify the properties of the retrieved DCB structure as appropriate.
         ' WARNING: Make sure to modify the properties according to their
supported values.
         MyDCB.BaudRate = 9600
         MyDCB.ByteSize = 8
         MyDCB.Parity = NOPARITY
         MyDCB.StopBits = ONESTOPBIT
         ' Reconfigure LPT1 based on the properties of the modified DCB
structure.
         Success = SetCommState(hParallelPort, MyDCB)
         If Success = False Then
            Throw New CommException("Unable to reconfigure LPT1")
         End If
         ' Retrieve the current time-out settings.
         Success = GetCommTimeouts(hParallelPort, MyCommTimeouts)
         If Success = False Then
            Throw New CommException("Unable to retrieve current time-out
settings")
         End If
         ' Modify the properties of the retrieved COMMTIMEOUTS structure as
appropriate.
         ' WARNING: Make sure to modify the properties according to their
supported values.
         MyCommTimeouts.ReadIntervalTimeout = 0
         MyCommTimeouts.ReadTotalTimeoutConstant = 0
         MyCommTimeouts.ReadTotalTimeoutMultiplier = 0
         MyCommTimeouts.WriteTotalTimeoutConstant = 0
         MyCommTimeouts.WriteTotalTimeoutMultiplier = 0
         ' Reconfigure the time-out settings, based on the properties of the
modified COMMTIMEOUTS structure.
         Success = SetCommTimeouts(hParallelPort, MyCommTimeouts)
         If Success = False Then
            Throw New CommException("Unable to reconfigure the time-out
settings")
         End If
         ' Write data to LPT1.
         ' Note: You cannot read data from a parallel port by calling the
ReadFile function.
         Console.WriteLine("Writing the following data to LPT1: Test")
         Success = WriteFile(hParallelPort, Buffer, Buffer.Length,
BytesWritten, IntPtr.Zero)
         If Success = False Then
            Throw New CommException("Unable to write to LPT1")
         End If
      Catch ex As Exception
         Console.WriteLine(Ex.Message)
      Finally
         ' Release the handle to LPT1.
         Success = CloseHandle(hParallelPort)
         If Success = False Then
            Console.WriteLine("Unable to release handle to LPT1")
         End If
      End Try

      Console.WriteLine("Press ENTER to quit")
      Console.ReadLine()
```

9. On the **Build** menu, click **Build Solution**.
10. On the **Debug** menu, click **Start** to run the application.

    You may receive the following text in the console:

    Accessing the COM1 serial port

Writing the following data to COM1: Test

Read the following data from COM1: *Serial Data*
Accessing the LPT1 parallel port

Writing the following data to LPT1: Test

Press ENTER to quit

Note *Serial Data* represents the data that you read from the serial port.

11. To close the application, press the ENTER key in the console.

Sample code listing (Module1.vb)

Before you use the following code example, replace *COM1* with the name of your serial port, and replace *LPT1* with the name of your parallel port.

Note The following code works only when serial devices and parallel devices are connected to the corresponding ports on the computer. If you do not connect these devices and you run the program, the program waits indefinitely.

```
Option Strict On

' Define a CommException class that inherits from the ApplicationException class.
' Then throw an object of type CommException when you receive an error message.
Class CommException
   Inherits ApplicationException
   Sub New(ByVal Reason As String)
      MyBase.New(Reason)
   End Sub
End Class

Module Module1
      'Declare structures
      Public Structure DCB
      Public DCBlength As Int32
      Public BaudRate As Int32
      Public fBitFields As Int32
      Public wReserved As Int16
      Public XonLim As Int16
      Public XoffLim As Int16
      Public ByteSize As Byte
      Public Parity As Byte
      Public StopBits As Byte
      Public XonChar As Byte
      Public XoffChar As Byte
      Public ErrorChar As Byte
      Public EofChar As Byte
      Public EvtChar As Byte
      Public wReserved1 As Int16 'Reserved; Do Not Use
   End Structure

   Public Structure COMMTIMEOUTS
      Public ReadIntervalTimeout As Int32
      Public ReadTotalTimeoutMultiplier As Int32
      Public ReadTotalTimeoutConstant As Int32
      Public WriteTotalTimeoutMultiplier As Int32
      Public WriteTotalTimeoutConstant As Int32
   End Structure

   'Declare constants.
   Public Const GENERIC_READ As Int32 = &H80000000
   Public Const GENERIC_WRITE As Int32 = &H40000000
   Public Const OPEN_EXISTING As Int32 = 3
   Public Const FILE_ATTRIBUTE_NORMAL As Int32 = &H80
   Public Const NOPARITY As Int32 = 0
   Public Const ONESTOPBIT As Int32 = 0

   'Declare references to external functions.
   Public Declare Auto Function CreateFile Lib "kernel32.dll" _
      (ByVal lpFileName As String, ByVal dwDesiredAccess As Int32, _
         ByVal dwShareMode As Int32, ByVal lpSecurityAttributes As IntPtr, _
            ByVal dwCreationDisposition As Int32, ByVal dwFlagsAndAttributes As Int32, _
               ByVal hTemplateFile As IntPtr) As IntPtr

   Public Declare Auto Function GetCommState Lib "kernel32.dll" (ByVal nCid As IntPtr, _
      ByRef lpDCB As DCB) As Boolean

   Public Declare Auto Function SetCommState Lib "kernel32.dll" (ByVal nCid As IntPtr, _
      ByRef lpDCB As DCB) As Boolean

   Public Declare Auto Function GetCommTimeouts Lib "kernel32.dll" (ByVal hFile As
IntPtr, _
      ByRef lpCommTimeouts As COMMTIMEOUTS) As Boolean

   Public Declare Auto Function SetCommTimeouts Lib "kernel32.dll" (ByVal hFile As
IntPtr, _
      ByRef lpCommTimeouts As COMMTIMEOUTS) As Boolean

   Public Declare Auto Function WriteFile Lib "kernel32.dll" (ByVal hFile As IntPtr, _
      ByVal lpBuffer As Byte(), ByVal nNumberOfBytesToWrite As Int32, _
         ByRef lpNumberOfBytesWritten As Int32, ByVal lpOverlapped As IntPtr) As Boolean

   Public Declare Auto Function ReadFile Lib "kernel32.dll" (ByVal hFile As IntPtr, _
      ByVal lpBuffer As Byte(), ByVal nNumberOfBytesToRead As Int32, _
         ByRef lpNumberOfBytesRead As Int32, ByVal lpOverlapped As IntPtr) As Boolean

   Public Declare Auto Function CloseHandle Lib "kernel32.dll" (ByVal hObject As IntPtr)
As Boolean

   Sub Main()

      ' Declare local variables that you will use in the code.
      Dim hSerialPort, hParallelPort As IntPtr
      Dim Success As Boolean
      Dim MyDCB As DCB
      Dim MyCommTimeouts As COMMTIMEOUTS
      Dim BytesWritten, BytesRead As Int32
      Dim Buffer() As Byte

      ' Declare variables to use for encoding.
      Dim oEncoder As New System.Text.ASCIIEncoding
      Dim oEnc As System.Text.Encoding = oEncoder.GetEncoding(1252)

      ' Convert String to Byte().
      Buffer = oEnc.GetBytes("Test")

      Try
         ' Serial port.
         Console.WriteLine("Accessing the COM1 serial port")
         ' Obtain a handle to the COM1 serial port.
         hSerialPort = CreateFile("COM1", GENERIC_READ Or GENERIC_WRITE, 0, IntPtr.Zero,
_
            OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, IntPtr.Zero)
         ' Verify that the obtained handle is valid.
         If hSerialPort.ToInt32 = -1 Then
            Throw New CommException("Unable to obtain a handle to the COM1 port")
         End If
         ' Retrieve the current control settings.
         Success = GetCommState(hSerialPort, MyDCB)
         If Success = False Then
            Throw New CommException("Unable to retrieve the current control settings")
         End If
         ' Modify the properties of the retrieved DCB structure as appropriate.
         ' WARNING: Make sure to modify the properties according to their supported
values.
         MyDCB.BaudRate = 9600
         MyDCB.ByteSize = 8
         MyDCB.Parity = NOPARITY
         MyDCB.StopBits = ONESTOPBIT
         ' Reconfigure COM1 based on the properties of the modified DCB structure.
         Success = SetCommState(hSerialPort, MyDCB)
         If Success = False Then
            Throw New CommException("Unable to reconfigure COM1")
         End If
         ' Retrieve the current time-out settings.
         Success = GetCommTimeouts(hSerialPort, MyCommTimeouts)
         If Success = False Then
            Throw New CommException("Unable to retrieve current time-out settings")
         End If
         ' Modify the properties of the retrieved COMMTIMEOUTS structure as appropriate.
         ' WARNING: Make sure to modify the properties according to their supported
values.
         MyCommTimeouts.ReadIntervalTimeout = 0
         MyCommTimeouts.ReadTotalTimeoutConstant = 0
         MyCommTimeouts.ReadTotalTimeoutMultiplier = 0
         MyCommTimeouts.WriteTotalTimeoutConstant = 0
         MyCommTimeouts.WriteTotalTimeoutMultiplier = 0
         ' Reconfigure the time-out settings, based on the properties of the modified
COMMTIMEOUTS structure.
         Success = SetCommTimeouts(hSerialPort, MyCommTimeouts)
         If Success = False Then
            Throw New CommException("Unable to reconfigure the time-out settings")
         End If
         ' Write data to COM1.
         Console.WriteLine("Writing the following data to COM1: Test")
         Success = WriteFile(hSerialPort, Buffer, Buffer.Length, BytesWritten,
IntPtr.Zero)
         If Success = False Then
            Throw New CommException("Unable to write to COM1")
         End If
         ' Read data from COM1.
         Success = ReadFile(hSerialPort, Buffer, BytesWritten, BytesRead, IntPtr.Zero)
         If Success = False Then
            Throw New CommException("Unable to read from COM1")
         End If
      Catch ex As Exception
         Console.WriteLine(Ex.Message)
      Finally
         ' Release the handle to COM1.
         Success = CloseHandle(hSerialPort)
         If Success = False Then
            Console.WriteLine("Unable to release handle to COM1")
         End If
      End Try

      Try
         ' Parallel port.
         Console.WriteLine("Accessing the LPT1 parallel port")
         ' Obtain a handle to the LPT1 parallel port.
         hParallelPort = CreateFile("LPT1", GENERIC_READ Or GENERIC_WRITE, 0,
IntPtr.Zero, _
            OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, IntPtr.Zero)
         ' Verify that the obtained handle is valid.
         If hParallelPort.ToInt32 = -1 Then
            Throw New CommException("Unable to obtain a handle to the LPT1 port")
         End If
         ' Retrieve the current control settings.
         Success = GetCommState(hParallelPort, MyDCB)
         If Success = False Then
            Throw New CommException("Unable to retrieve the current control settings")
         End If
         ' Modify the properties of the retrieved DCB structure as appropriate.
         ' WARNING: Make sure to modify the properties according to their supported
values.
         MyDCB.BaudRate = 9600
         MyDCB.ByteSize = 8
         MyDCB.Parity = NOPARITY
         MyDCB.StopBits = ONESTOPBIT
         ' Reconfigure LPT1 based on the properties of MyDCB.
         Success = SetCommState(hParallelPort, MyDCB)
```

```
            If Success = False Then
               Throw New CommException("Unable to reconfigure LPT1")
            End If
            ' Reconfigure LPT1 based on the properties of the modified DCB structure.
            Success = GetCommTimeouts(hParallelPort, MyCommTimeouts)
            If Success = False Then
               Throw New CommException("Unable to retrieve current time-out settings")
            End If
            ' Modify the properties of the retrieved COMMTIMEOUTS structure as appropriate.
            ' WARNING: Make sure to modify the properties according to their supported
values.
            MyCommTimeouts.ReadIntervalTimeout = 0
            MyCommTimeouts.ReadTotalTimeoutConstant = 0
            MyCommTimeouts.ReadTotalTimeoutMultiplier = 0
            MyCommTimeouts.WriteTotalTimeoutConstant = 0
            MyCommTimeouts.WriteTotalTimeoutMultiplier = 0
            ' Reconfigure the time-out settings, based on the properties of the modified
COMMTIMEOUTS structure.
            Success = SetCommTimeouts(hParallelPort, MyCommTimeouts)
            If Success = False Then
               Throw New CommException("Unable to reconfigure the time-out settings")
            End If
            ' Write data to LPT1.
            ' Note: You cannot read data from a parallel port by calling the ReadFile
function.
            Console.WriteLine("Writing the following data to LPT1: Test")
            Success = WriteFile(hParallelPort, Buffer, Buffer.Length, BytesWritten,
IntPtr.Zero)
            If Success = False Then
               Throw New CommException("Unable to write to LPT1")
            End If
         Catch ex As Exception
            Console.WriteLine(Ex.Message)
         Finally
            ' Release the handle to LPT1.
            Success = CloseHandle(hParallelPort)
            If Success = False Then
               Console.WriteLine("Unable to release handle to LPT1")
            End If
         End Try

         Console.WriteLine("Press ENTER to quit")
         Console.ReadLine()

      End Sub

End Module
```

Troubleshoot

- When you run the application, you may receive the following error message:

  System.NullReferenceException: Object reference not set to an instance of an object.

  You may receive this error message because your function declarations are incorrect. This error message typically occurs when your declarations contain **ByVal** parameters instead of **ByRef** parameters.
- Your application may wait indefinitely when you invoke the **ReadFile** function. This behavior typically occurs when you set the read time-outs to zero in the retrieved **COMMTIMEOUTS** structure. To resolve this issue, modify the properties of the **COMMTIMEOUTS** structure, as appropriate.

REFERENCES

For more information, visit the following Microsoft Developer Network (MSDN) Web sites:

Communications resources
http://msdn2.microsoft.com/en-us/library/aa363196.aspx (http://msdn2.microsoft.com/en-

us/library/aa363196.aspx)

Interoperating with unmanaged code
http://msdn2.microsoft.com/en-us/library/sd10k43k(vs.71).aspx (http://msdn2.microsoft.com/en-us/library/sd10k43k(vs.71).aspx)

System.Runtime.InteropServices namespace
http://msdn2.microsoft.com/en-us/library/system.runtime.interopservices(vs.71).aspx (http://msdn2.microsoft.com/en-us/library/system.runtime.interopservices(vs.71).aspx)

DllImportAttribute class
http://msdn2.microsoft.com/en-us/library/system.runtime.interopservices.dllimportattribute(vs.71).aspx (http://msdn2.microsoft.com/en-us/library/system.runtime.interopservices.dllimportattribute(vs.71).aspx)

The Windows API and other dynamic-link libraries
http://msdn2.microsoft.com/en-us/library/aa141322(office.10).aspx (http://msdn2.microsoft.com/en-us/library/aa141322(office.10).aspx)

Understanding handles
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/modcore/html/deovrUnderstandingHandles.asp (http://msdn2.microsoft.com/en-us/library/aa141354(office.10).aspx)

For more information about AT modem commands, click the following article number to view the article in the Microsoft Knowledge Base:

164660 (http://support.microsoft.com/kb/164660/) AT modem command reference

APPLIES TO

- Microsoft Visual Basic .NET 2003 Standard Edition
- Microsoft Visual Basic .NET 2002 Standard Edition

Keywords: kbvs2005swept kbvs2005doesnotapply kbserial kbsdk kbkern32dll kbdll kbcommport kbapi kb32bitonly kbinterop kbconsole kbconfig kbappdev kbprogramming kbsample kbhowtomaster KB823179